ECAN527HR Isolated 1 Mb/s CAN interface board User's Manual

BDM-610020019
Rev. A

ECAN527HR ISOLATED 1 MB/S CAN INTERFACE BOARD USER'S MANUAL

RTD Embedded Technologies, INC.
103 Innovation Blvd.
State College, PA 16803-0906

Phone: +1-814-234-8087
FAX: +1-814-234-5218

E-mail
sales@rtd.com
techsupport@rtd.com

web site
http://www.rtd.com

Revision History

Rev. A New Manual Naming Method

Published by:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College, PA 16803-0906

Copyright 1999, 2002, 2003 by RTD Embedded Technologies, Inc.
All rights reserved
Printed in U.S.A.

The RTD Logo is a registered trademark of RTD Embedded Technologies. cpuModule and utilityModule are trademarks of RTD Embedded Technologies. PhoenixPICO and PheonixPICO BIOS are trademarks of Phoenix Technologies Ltd. PS/2, PC/XT, PC/AT and IBM are trademarks of International Business Machines Inc. MS-DOS, Windows, Windows 95, Windows 98 and Windows NT are trademarks of Microsoft Corp. PC/104 is a registered trademark of PC/104 Consortium. All other trademarks appearing in this document are the property of their respective owners.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 9

FEATURES OF THE ECAN527HR INCLUDE: 9
CAN-BUS CONTROLLER 9
PHYSICAL INTERFACE 9
MECHANICAL DESCRIPTION 10
CONNECTOR DESCRIPTION 10
WHAT COMES WITH YOUR BOARD 10
BOARD ACCESSORIES 11
USING THIS MANUAL 11
WHEN YOU NEED HELP 11
BOARD SETTINGS 11
BASE ADDRESS JUMPERS (FACTORY SETTING: D0000 H) 13

CHAPTER 2 BOARD INSTALLATION 17

BOARD INSTALLATION 17
General installation guidelines: *17*
Installation integrated with a PC/104 module stack: *17*
External I/O connections *19*
General purpose digital I/O 19
Galvanically isolated CAN bus connector 20

CHAPTER 3 HARDWARE DESCRIPTION 22

Galvanic isolation of the CAN-bus *24*
Fiberoptic interface *24*
Configuration EEPROM *25*
Digital I/O *25*

CHAPTER 4 BOARD OPERATION AND PROGRAMMING 26

DEFINING THE MEMORY MAP 26
Bit timing registers *26*
INTERRUPTS 26
What is an interrupt ? *26*
Interrupt request lines *27*
8259 Programmable Interrupt Controller *27*
Interrupt Mask Register (IMR) *27*
End-of-Interrupt (EOI) Command *27*
What exactly happens when an interrupt occurs? *27*
Using Interrupt in your Program *28*
Writing an Interrupt Service Routine (ISR) *28*
Your ISR should have the following structure: *29*
Saving the Startup Interrupt Mask Register (IMR) and interrupt vector *30*
Common Interrupt mistakes *30*
Example on Interrupt vector table setup in C-code: *31*

APPENDIX A 32

ECAN527HR SPECIFICATIONS 32
Host Interface *32*
CAN Interface *32*
using the RTD CAN Spider active hub on ECAN527HR-2 boards *32*
Digital I/O (non isolated) 32
Connectors 32
Electrical 33

CE 33

CHAPTER 5 RETURN POLICY AND WARRANTY 34

RETURN POLICY 34
LIMITED WARRANTY 35

LIST OF ILLUSTRATIONS AND TABLES

FIGURES

FIG. 1 - BOARD LAYOUT SHOWING JUMPER LOCATIONS 12
FIG. 2 BASE ADDRESS JUMPERS ILLUSTRATING D8000H BASE ADDRESS 15
FIG. 3 INTERRUPT JUMPERS SET TO IRQ 5 15
FIG. 4 ECAN527HR INTEGRATED IN A PC/104 DATAMODULE STACK 18
FIG. 5 19" EUROCARD RACK INSTALLATION WITH AN INTEGRATED PC/104 DATAMODULE AND EUROCARD CPUMODULE COMPUTER SYSTEM 18
FIG. 6 ECAN527HR BLOCK DIAGRAM 22
FIG. 7 CAN-BUS PHYSICAL INTERFACE CIRCUITRY, POWER SUPPLY OMITTED FOR CLARITY 24

TABLES

TABLE 1FACTORY CONFIGURED JUMPER SETTINGS.(SEE FIGURE 1-1 FOR DETAILED LOCATIONS) 12
TABLE 2 BASE ADDRESS JUMPER SETTINGS 14
TABLE 3DIGITAL I/O CONNECTOR OF THE ECAN527HR 19
TABLE 4 PHYSICAL INTERFACE CONNECTOR PINOUTS OF THE ECAN527HR 20
TABLE 5DIGITAL I/O CONFIGURATION OF PORT P2 OF THE 82527 CHIP 25

Chapter 1 INTRODUCTION

This user's manual describes the operation of the ECAN527HR CAN-bus interface board.

Features of the ECAN527HR include:

- 1 Mb/s maximum datarate (fully programmable)
- Full CAN-functionality 2.0 B
- Intel 82527 CAN-bus controller
- 256 bytes of configuration EEPROM
- Galvanically isolated physical interface with CAN transceiver
- Optional fiberoptic transmitter and receiver for noisy environments
- 6 user accessible bit programmable bidirectional digital I/O
- 2 status LED's
- +5V only operation
- PC/104 compliant

The following paragraphs briefly describe the major features of the ECAN527HR. A more detailed discussion is included in Chapter 3 (Hardware description) and in Chapter 4 (Board operation and programming). The board set-up is described in Chapter 1 (Board Settings). A full description of the Intel 82527 CAN-controller is included in Chapter 4.

CAN-bus controller

The ECAN527HR CAN-bus interface is implemented using the Intel 82527 chip. This controller supports CAN Specification 2.0. This versatile chip supports standard and extended data and remote frames as follows; A programmable global message identifier mask; 15 message objects of 8-byte data length and a programmable bit rate. This fully integrated chip supports all the functionality of the CAN-bus protocol.

Physical Interface

Industrial environments require galvanic isolation and bus filtering to provide reliable data communication and safety. The ECAN527HR has two options for the physical interface.

1) The *galvanivally isolated physical interface* is implemented using high speed optocouplers and a DC/DC converter. To protect the input from radiated bus noise a special balanced bus filter is used. This filter is designed to meet EMI requirements.

2) The *fiberoptic physical interface* is implemented using HP Versa-Link fiberoptic transmitter and receiver. The fiberptic interface may be used for point to point CAN networks. Due to the fiberoptic technology, bitwise arbitration of the CAN bus protocol can not be supported. You should use the CANSPIDER active hub available from RTD to build fiberoptic CAN systems.

Mechanical description

The ECAN527HR is designed on a PC/104 form factor. An easy mechanical interface to both PC/104 and EUROCARD systems can be achieved. Stack your ECAN527HR directly on a PC/104 compatible computer using the onboard mounting holes.

Connector description

There is a 10-pin interface connector on the ECAN527HR which directly interfaces to the galvanivally isolated CAN bus.

The fiberoptic interface may directly connect to fiberoptic cable using HP Versa-Link PCB-mounted connectors.

The general purpose digital outputs and inputs are connected to the ECAN527HR by 10-pin header connector.

What comes with your board

You will receive the following items in your ECAN527HR package:

* ECAN527HR CAN bus interface module
* User's manual

Note : Software and drivers are available on our website at www.rtd.com.

If any item is missing or damaged, please call RTD Embedded Technologies, Inc.customer service department at (814) 359-3174.

Board accessories

In addition to the items included in your ECAN527HR delivery, several software and hardware accessories are available. Call your distributor for more information on these accessories and for help in choosing the best items to support your distributed control system.

* Application software

Third party high level protocol drivers
QNX drivers
CanOpen libraries for QNX and LINUX

* Hardware accessories

CANSPIDER Active hub for fiberoptical networking using ECAN527-2
IDAN ruggedized aliminium frames and enclosures for PC/104 systems (IDAN-ECAN527HR)

Using this manual

This manual is intended to help you install your new ECAN527HR card and get it running quickly, while also providing enough detail about the board and it's functions so that you can enjoy maximum use of it's features even in the most demanding applications.

When you need help

This manual and all the example programs will provide you with enough information to fully utilize all the features on this board. If you have any problems installing or using this board,

contact our Technical Support Department (814) 234-8087 during European business hours, or send a FAX to (814) 234-5218 or Email to **sales@rtd.com**. When sending

a FAX or Email request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

BOARD SETTINGS

The ECAN527HR CAN bus interface board has jumper settings which can be changed to suit your application and host computer memory configuration. The factory settings are listed and shown in the diagram in the beginning of this chapter.

Factory configured Jumper Settings

Table 1 illustrates the factory jumper setting for the ECAN527HR. Figure 1 shows the board layout of the ECAN527HR and the locations of the jumpers. The following paragraphs explain how to change the factory jumper settings to suit your specific application.

Table 1Factory configured jumper settings.(See figure 1-1 for detailed locations)

Jumper Name	JUMPER DESCRIPTION	NUMBER OF JUMPERS	FACTORY DEFAULT
ADDR	BASE ADDRESS	11	D0000
IRQ	HOST INTERRUPT	10	5
J34	CAN-BUS TERMINATION	1	CLOSED

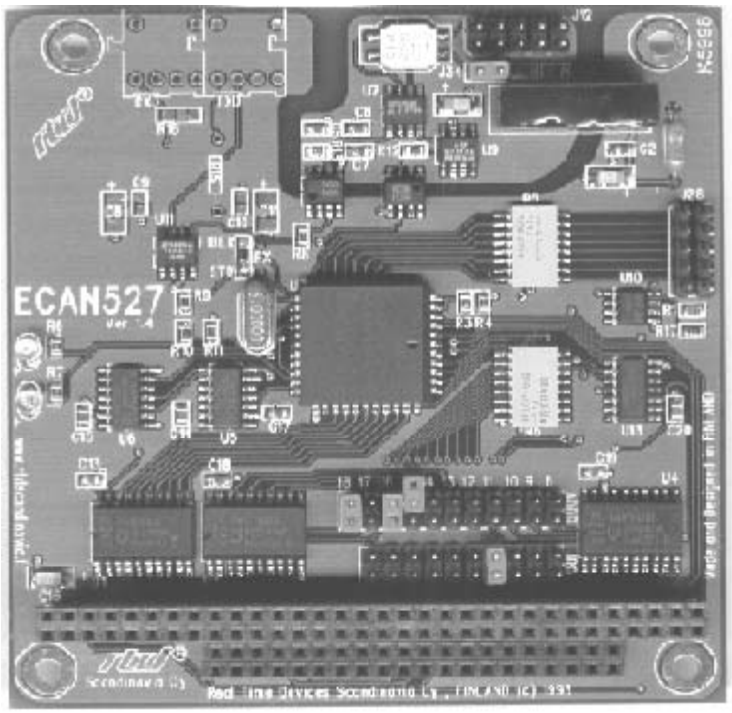

Fig. 1 - Board layout showing jumper locations

Base Address Jumpers (Factory setting: D0000 h)

The ECAN527HR is memory mapped into the low memory of your host XT/AT. The ECAN527HR board occupies a memory window of 256 bytes starting from the base address.

The most common cause of failure when you are first setting up your module is address contention. Some of your computers memory space is already occupied by other devices and memory resident programs. When the ECAN527HR attempts to use reserved memory addresses already used by another device, erratic performance may occur and data read from the board may be corrupted.

To avoid this problem make sure you set up the base address first using the 11 jumpers

marked "ADDR" which let you choose from number of addresses in your host computers memory map. Should the factory installed setting of D0000h be unusable for your system configuration, you may change this setting to another using the options illustrated in Table 2. The table shows the jumper settings and their corresponding values in hexadecimal values. Make sure you verify the correct location of the base address jumpers. When the jumper is removed it corresponds to a logical "0", connecting the jumper to a "1". When you set the base address of the module, record the setting in the table inside the back cover of this manual after the Appendices.

Note: If you are using a memory manager such as QEMM, make sure you exclude the

---

memory section you are occupying by the ECAN527; for example X=D0000-D00FF.

Address Jumper Settings ECAN527HR	
Base Address Hex	Jumper Settings 18 17 16 15
80XXX	0 0 0 0
88XXX	0 0 0 1
90XXX	0 0 1 0
98XXX	0 0 1 1
A0XXX	0 1 0 0
A8XXX	0 1 0 1
B0XXX	0 1 1 0
B8XXX	0 1 1 1
C0XXX	1 0 0 0
C8XXX	1 0 0 1
D0XXX	1 0 1 0
D8XXX	1 0 1 1
E0XXX	1 1 0 0
E8XXX	1 1 0 1
F0XXX	1 1 1 0
F8XXX	1 1 1 1
0 = NO JUMPER, 1 = JUMPER installed Note: A19 is always decoded as 1!	

Table 2 Base address jumper settings

Note: The above table illustrates only the settings for the high address bits A18-A15. If you for instance want to configure address D0E00 - D0EFF for your ECAN527HR you must set the jumpers as: 1 0 1 _ 0 0 0 0 _ 1 1 1 0 ! Address line A19 is always decoded as "1".

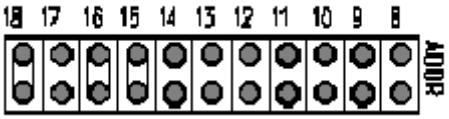

Fig. 2 Base address jumpers illustrating D8000h base address

Interrupt Channel (Factory setting: IRQ 5)

The header connector, shown in Figure 3, lets you connect the onboard 82527 CAN controller interrupt output to one of the interrupt channels available on the host XT/AT- bus. If your board has no AT-bus extension interrupts 10-15 are not available.

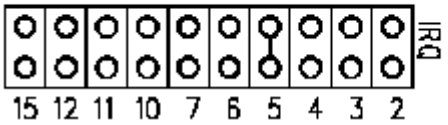

Fig. 3 Interrupt jumpers set to IRQ 5

Note: The ECAN527HR hardware revision 1.4 onward do not support interrupt sharing.

Chapter 2 BOARD INSTALLATION

The ECAN527HR CAN-bus interface board is very easy to connect to your industrial distributed control system. Direct interface to PC/104 systems as well as EUROCARD boards is achieved. This chapter tells you step-by-step how to install the ECAN527HR into your system.

After completing the installation use the diagnostic and test software to fully verify that your board is working. The latest versions of the test software as well drivers are available on our website at <www.rtd.com>.

Board Installation

Keep your board in its antistatic bag until you are ready to install it to your system! When removing it from the bag, hold the board at the edges and do not touch the components or connectors. Please handle the board in an antistatic environment and use a **grounded** workbench for testing and handling of your hardware.

Before installing the board in your computer, check the jumper settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable board operation and erratic response.

General installation guidelines:

1. Turn OFF the power to your computer and all devices connected to ECAN527
2. Touch the grounded metal housing of your computer to discharge any antistatic buildup and then remove the board from its antistatic bag.
3. Hold the board by the edges and install it in an enclosure or place it on the table on an antistatic surface.
4. Connect the board to the CAN fieldbus using the two fieldbus interface header connectors. Make sure the polarity of the cable is correct. Both connectors are identical, one of these headers may be used to bring the CAN bus to the board; the other connector interfaces your ECAN527HR to the CAN bus.

Installation integrated with a PC/104 module stack:

* Secure the four PC/104 installation holes with standoffs.
* Connect the board to the CAN bus using the two fieldbus interface header connectors, alternatively use the fiberoptic connectors.

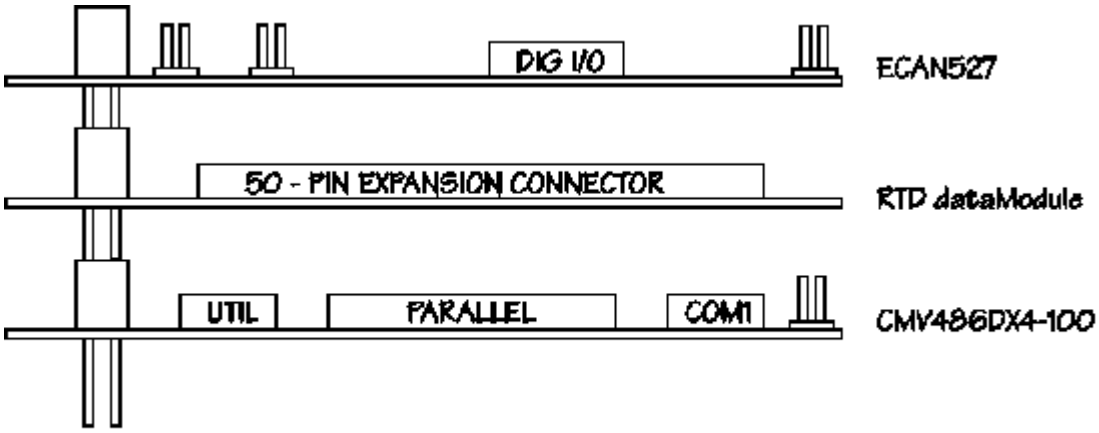

Fig. 4 ECAN527HR integrated in a PC/104 dataModule stack

3U rack or enclosure installation with a EUROCARD CPU containing one ECAN527.

* The PC/104 system can easily be inserted into a 19" rack installation using the CPU as a "form factor adapter". Assemble your PC/104 data modules on a RTD single board computer and install the system in an 19" enclosure. Multiple ECAN527HR boards can easily be connected to this system. See figure 5 below.

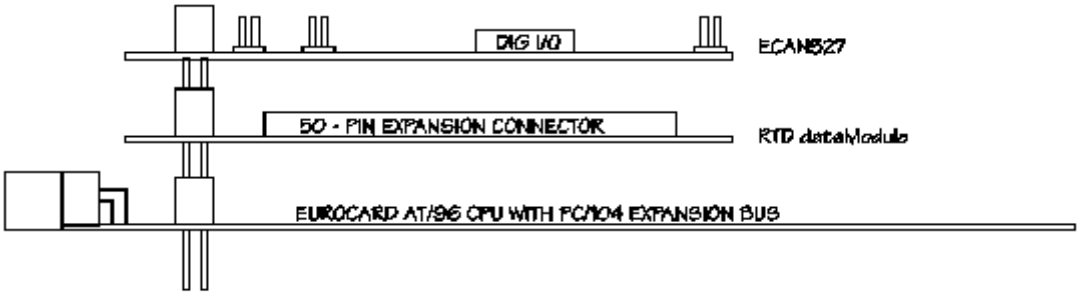

Fig. 5 19" Eurocard rack installation with an integrated PC/104 dataModule and EUROCARD cpuModule computer system

External I/O connections

General purpose digital I/O

Table 2-1 shows the connector layout of the ECAN527HR digital I/O connector J28. This connector is located toward the left side of the board. Refer to this diagram when making signal connections.

Pin Number	Function
1	I/O_1 /LED1
2	I/O_2 /LED2
3	EEPROM clk
4	EEPROM data
5	I/O_3
6	I/O_4
7	I/O_5
8	I/O_6
9	+5V
10	GND

Table 3Digital I/O connector of the ECAN527HR

Note: I/O signals 3 and 4 are reserved for the use of the non volatile EEPROM. In this case these pins are not available for user I/O.

Galvanically isolated CAN bus connector

Table 4 shows the CAN physical interface connector. This connectors is used to route the CAN-bus through your ECAN527HR board. It is located at the top of your board. This connector conforms to the ISO11898 specification.

Pin Number	Function	Pin Number	Function
1	NC	2	ISO_GND
3	BUS_L	4	BUS_H
5	ISO_GND	6	NC
7	NC	8	ISO+5V
9	ISO_GND	10	NC

Table 4 Physical interface connector pinouts of the ECAN527HR

Chapter 3 HARDWARE DESCRIPTION

Chapter 3 - **Hardware Description** describes the major features of the ECAN527HR: the Intel 82527 CAN bus controller, Galvanic isolation of the CAN-bus, the Fiberoptic interface, the Onboard configuration EEPROM, and Digital I/O.

Figure 6 shows the general block diagram of the ECAN527HR. This chapter describes the major features of the ECAN527HR: the Intel 82527 CAN-bus controller, Galvanic isolation of the CAN-bus, the Fiberoptic interface, the Onboard configuration EEPROM, and Digital I/O.

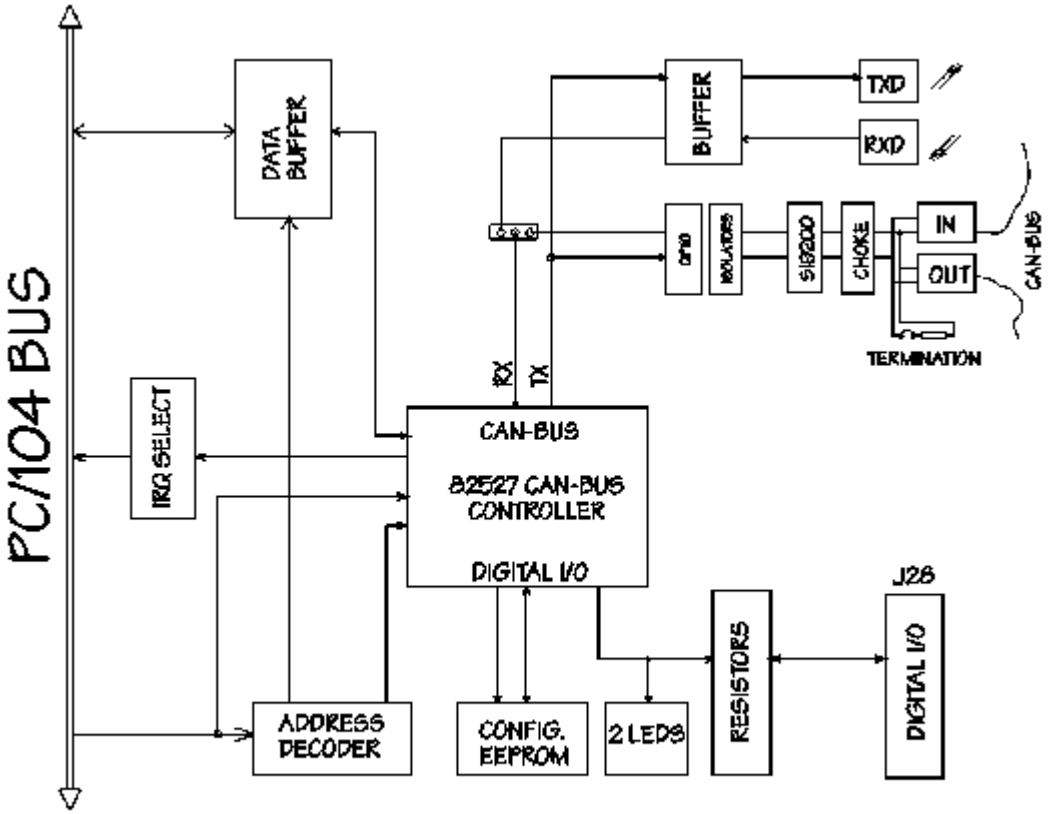

Fig. 6 ECAN527HR Block diagram

82527 CAN bus controller Reference note:

(Intel publication CAN Architectural Overview, Automotive Products Databook)

The 82527 CAN controller consists of six functional blocks. The CPU interface logic manages the communication to the host computer. The CAN controller interfaces to the CAN bus and implements the protocol rules of the CAN protocol for the transmission and reception of messages. The RAM is the physical interface layer between the host CPU and the CAN bus. One eight bit I/O port provides low speed I/O capabilities.

The 82527 RAM provides storage for 15 message objects of 8 byte length. Each message object has a unique identifier and can be configured to either transmit of to receive except for the last message object. The last message object is a receive- only buffer with a special mask design to allow selected groups of different message identifiers to be received.

Each message identifier contains control and status bits. A message object with a direction set for receive will send a remote frame by requesting a message transmission. A message set as transmit will be configured to automatically send a data frame whenever a remote frame with a matching identifier is received over the CAN bus. All message objects have separate transmit and receive interrupts and status bits, allowing the CPU full flexibility in detecting when a remote frame has been sent or received.

The 82527 chip supports a global masking feature for acceptance filtering. This feature allows the user to globally mask, or "don't care", any identifier bits of the incoming message.

This mask is programmable to allow the user to design an application specific message identification strategy. There are separate global mask for standard and extended frames.

The incoming message first passes through the global mask and is matched to the identifiers in the message objects 1-14. If there is no identifier match then the message passes through the local mask in message object 15. The local mask allows a large number of infrequent messages to be received by the 82527. Message object 15 is also buffered to allow the CPU time

to service a message received.

Galvanic isolation of the CAN-bus

The galvanic isolation of the ECAN527HR is implemented using 1) Optocouplers for data transmission and 2) a DC/DC converter to supply power to the CAN bus and the physical interface circuitry.

The high speed optocouplers are directly connected to the 82527. The optocouplers drive the CAN bus transceiver. A special balanced CAN bus choke is used not only to improve immunity to bus noise, but also to protect the bus transceiver. This filter assists in conforming to the EN55022 Radiated Emission test requirements.

A 1W DC/DC converter may be used to power other remote devices on the CAN bus. The output power of this converter is isolated up to 1,5 kV peak.

Figure 7 illustrates the physical interface circuitry of the galvanically isolated CAN-bus.

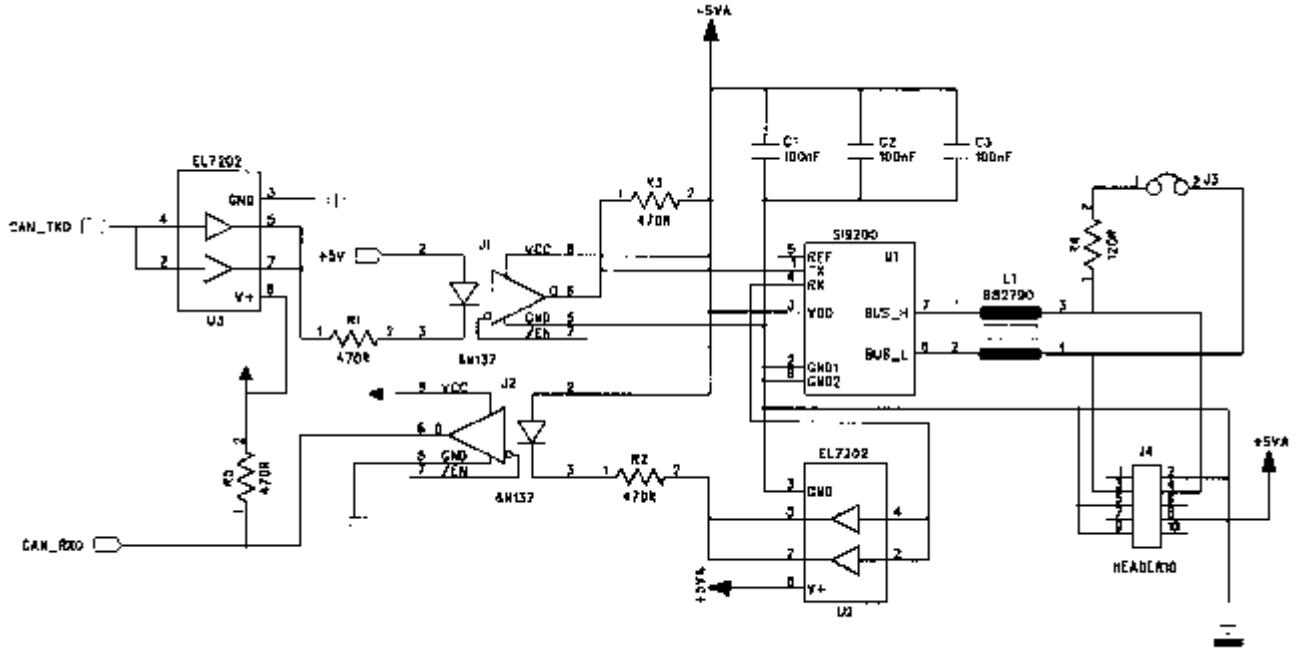

Fig. 7 CAN-bus physical interface circuitry, power supply omitted for clarity

Fiberoptic interface

An optional fiber optic interface can be used to implement point to point communication using the CAN protocol. Due to the physics of the fiberoptic media, bitwise arbitration featured by the CAN protocol can not be supported. The fiberoptic interface may be used in environments where either immunity to long distances or excessive noise is a design criteria. It is possible to combine wire and fiberoptic connections in the same network using active repeater nodes such as the RTD CAN Spider active 4-port CAN hub with multiple ECAN527HR boards.

The fiberoptic interface supports datarates up to 1 Mb/s using Hewlett Packard Versa-Link transmitter and receiver modules. These modules enable direct connectivity to fiberoptic cables from your ECAN527HR. The CAN Spider active hub also features compatible transmitters and receivers.

Configuration EEPROM

A configuration EEPROM may store up to 256 bytes of user data. This memory may be used to store CAN bus parameters or other identifier codes required by the system. The EEPROM used the I2C standard interfacing scheme. The address of the EEPROM device is set to 000. To communicate to the EEPROM you must drive the clock and data lines of the device through the port P2.2-3 lines of the 82527 CAN controller.

The configuration EEPROM is interfaced through the digital I/O lines of port P2 bits 2 and 3 (LSB is 0). Data is read and written by clocking these lines to provide the serial interface required by the EEPROM device.

Digital I/O

Table 5 shows the functionality of the 82527 port P2 bits. Bits 0 and 1 have two functionalities. They may either drive the two onboard status LED's, or operate as user TTL-level I/O. The output impedance of this port is 100 Ohms. This series resistor on every I/O line provides protection against short circuits and transient voltages.

P2.0	P2.1	P2.2	P2.3	P2.4	P2.5	P2.6	P2.7
LED1	LED2	CLK	DATA	DIO3	DIO4	DIO5	DIO6

Table 5Digital I/O configuration of port P2 of the 82527 chip

Chapter 4 BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program and use your ECAN527HR. It provides a complete detailed description of the memory map and a detailed discussion of programming operations to aid you in programming. The full functionality of the ECAN527HR is described in the datasheet reprint from Intel on the 82527 CAN controller chip.

Defining the Memory Map

The memory map of the ECAN527HR occupies 256 bytes of host PC low memory space. This window is freely selectable by the user as described in Chapter 1 (table 2). After setting the base address you have access to the internal resources of the 82527 CAN controller chip, as is described in the next sections reprinted from the Intel 82527 manual.

Bit timing registers

These are the recommended bit timing register values for reliable communication at 1MBps, for 500Kbps change BIT_TIM0 to 01h.

BIT_TIM0	**00h**
BIT_TIM1	**C6h**

For more information on these and other control registers of the CAN controller chip please refer to the component specific datasheet reprinted from Intel attached.

Interrupts

What is an interrupt ?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are a very flexible way of dealing with asynchronous events. Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time to do nothing while waiting for a keystroke to occur. Thus the interrupt scheme is used and the processor proceeds with other tasks. When a keystroke occurs, the keyboard 'interrupts' the processor , the processor then gets the keyboard data which is then placed into the memory.It then returns to what it was doing before the interrupt occurred. Other common devices that use interrupts are network boards, A/D boards, serial ports etc.

Your ECAN527HR can interrupt the main processor when a message is received or transmitted if interrupts are enabled on the ECAN527HR board. By using interrupts you can write powerful code to interface to your CAN network.

Interrupt request lines

To allow different peripheral devices to generate interrupts on the same computer, the PC AT bus has interrupt request channels (IRQ's). A rising edge transition on one of these lines will be latched into the interrupt controller. The interrupt controller checks to see if the interrupts are to be acknowledged from that IRQ and, if another interrupt is being processed, it decides if the new request should supercede the one in progress or if it has to wait until the one in progress is done. The priority level of the interrupt is determined by the number of the IRQ; IRQ0 has the highest priority IRQ15 the lowest. Many of the IRQ's are used by the standard system resources. IRQ0 is dedicated for the internal timer, IRQ1 is dedicated to the keyboard input, IRQ3 for serial port COM2 and IRQ4 for serial port COM1. Often interrupts 3,5 and 7 are free for the user.

8259 Programmable Interrupt Controller

The chip responsible for handling interrupt requests in a PC is the 8259 Interrupt Controller. To use interrupts you will need to know how to read and set the 8259's internal interrupt mask register (IMR) and how to send the end-of-interrupt (EOI) command to acknowledge the 8259 interrupt controller.

Interrupt Mask Register (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of the interrupt line. If a bit is set (equal to 1), then the corresponding IRQ is masked, and it will not generate an interrupt. If a bit is cleared (equal to 0), then the corresponding IRQ is not masked, and it can then generate an interrupt. The interrupt mask register is programmed through **port 21h**.

End-of-Interrupt (EOI) Command

After an interrupt service routine is complete, the 8259 Interrupt Controller must be acknowledged by **writing the value 20h to port 20h**.

What exactly happens when an interrupt occurs?

Understanding the sequence of events when an interrupt is triggered is necessary to correctly write interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the ECAN527), the interrupt controller checks to see if interrupts are enabled for that IRQ, and then checks to see if other interrupts are active or requested and determines which interrupt has priority. The interrupt controller then interrupts the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed onto the system stack, and a new set of CS and IP are loaded from the lowest 1024 bytes of memory.

This table is referred to as the interrupt vector table and each entry to this table is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table , the processor starts to execute code from the new Code Segment (CS) and from the new Instruction Pointer (IP). When the interrupt routine is completed the old CS and IP are popped from the system stack and the program execution continues from the point it was interrupted.

Using Interrupt in your Program

Adding interrupt support to your program is not as difficult as it may seem, especially when programming under DOS. The following discussion will cover programming under DOS. Note, that even the smallest mistake in your interrupt program may cause the computer to hang up and will only restart after a reboot. This can be frustrating and time-consuming.

Writing an Interrupt Service Routine (ISR)

The first step in adding interrupts to your software is to write an interrupt service routine (ISR). This is the routine that will be executed automatically each time an interrupt request occurs for the specified IRQ. An ISR is different from other subroutines or procedures. First, on entrance the processor registers must be pushed onto the stack before anything else! Second, just before exiting the routine , you must clear the interrupt on the ECAN527HR by writing to the 82527 CAN controller, and write the EOI command to the interrupt controller. Finally, when exiting the interrupt routine the processor registers must be popped from the system stack and you must execute the IRET assembly instruction. This instruction pops the CS, IP and processor flags from the system stack. These were pushed onto the stack when entering the ISR.

Most compilers allow you to identify a function as an interrupt type and will automatically add these instructions to your ISR with one exception: most compilers do not automatically add the EOI command to the function, you must do it yourself. Other than this and a few exceptions discussed below, you can write your ISR as any code routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend you stick to the basics; just something that enables you to verify you have entered the ISR and executed it successfully. For example: set a flag in your ISR and in your main program check for the flag.

Note: If you choose to write your ISR in in-line Assembly , you must push and pop registers correctly, and exit the routine with the IRET instruction instead of the RET instruction.

There are a few precautions you must consider when writing ISR's. The most important is, **do not use any DOS functions or functions that call DOS functions from an interrupt routine.** DOS is not reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about using interrupts? Then, you could have the situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to the DOS function X, then function X is essentially being called while active. Such cases will cause the computer to crash. DOS does not support such operation. A general rule is, that do not call any functions that use the screen, read keyboard input and any file I/O routines should not be used in ISR's.

The same problem of reentrancy exists for many floating point emulators as well, meaning you should avoid floating point mathematical operations in your ISR.

Note, that the problem of reentrancy exists, no matter what programming language you use. Even, if you are writing your ISR in Assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways to avoid this problem, such as those which involve checking if any DOS functions are currently active when your ISR is called, but such solutions are beyond the scope of this manual.

The second major concern when writing ISR's is to make them as short as possible in term of execution time. Spending long times in interrupt service routines may mean that other important interrupts are not serviced. Also, if you spend too long in your ISR, it may be called again before you have exited. This will lead to your computer hanging up and will require a reboot.

Your ISR should have the following structure:

- Push any processor registers used in your ISR. Most C compiler do this automatically
- Put the body of your routine here
- Read interrupt status register of the 82527 chip on your ECAN527HR board
- Clear the interrupt bit by writing to the 82527 CAN controller
- Issue the EOI command to the 8259 by writing 20h to address 20h
- Pop all registers. Most C compilers do this automatically

The following C example shows what the shell of your ISR should be like:

```
/*-------------------------------------------------------------------------------
| Function:         new_IRQ_handler
| Inputs:  Nothing
| Returns:          Nothing - Sets the interrupt flag for the EVENT.
|-------------------------------------------------------------------------------*/
void interrupt far new_IRQ_handler(void)
{

        IRQ_flag = 1;               // Indicate to main process interrupt has occurred
        {
        // Your program code should be here
        }
                                    // Read interrupt status registers
                                    // Clear interrupt on ECAN527
        outp(0x20, 0x20);           /* Acknowledge the interrupt controller. */
}
```

Saving the Startup Interrupt Mask Register (IMR) and interrupt vector

The next step after writing the ISR is to save the startup state of the interrupt mask register (IMR) and the original interrupt vector you are using. The IMR is located in address **21h**. The interrupt vector you will be using is located in the interrupt vector table which is an array of 4-byte pointers (addresses) and it is locate din the first 1024 bytes of the memory (Segment 0 offset 0). You can read this value directly, but it is a better practice to use DOS function 35h (get interrupt vector) to do this. Most C compilers have a special function available for doing this. The vectors for the hardware interrupts on the XT - bus are vectors 8-15., where IRQ0 uses vector 8 and IRQ7 uses vector 15. Thus if your ECAN527HR is using IRQ5 it corresponds to vector number 13.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read the current IMR at I/O port 21h, and set the bit that corresponds to tout IRQ. The IMR is arranged so that bit 0 is for IRQ0 and bit 7 is for IRQ7. See the paragraph entitled *Interrupt Mask Register (IMR*) earlier in this discussion for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21h.

with the startup IMR saved and the interrupts temporarily disabled, you can assign the interrupt vector to point to your ISR. Again you can overwrite the appropriate entry in the vector table with a direct memory write, but this is not recommended. Instead use the DOS function 25h (Set Interrupt Vector) or, if your compiler provides it, the library routine for setting up interrupt vectors. Remember that interrupt vector 8 corresponds to IRQ0, vector 9 for IRQ1 etc.

If you need to program the source of your interrupts, do that next. For example, if you are using transmitted or received messages as an interrupt source , program it to do that.

Finally, clear the mask bit for your IRQ in the IMR. This will enable your IRQ.

Common Interrupt mistakes

- Remember, hardware interrupts are from 8-15, XT IRQ's are numbered 0-7
- Forgetting to clear the IRQ mask bit in the IMR
- Forgetting to send the EOI command after ISR code. Disables further interrupts.

Example on Interrupt vector table setup in C-code:

```
void far _interrupt new_IRQ1_handler(void );          /* ISR function prototype */
#define IRQ1_VECTOR   3                               /* Name for IRQ */
void (interrupt far *old_IRQ1_dispatcher)
        (es,ds,di,si,bp,sp,bx,dx,cx,ax,ip,cs,flags);  /* Variable to store old IRQ_Vector */
void far _interrupt new_IRQ1_handler(void );

/*---------------------------------------------------------------------
| Function:         init_irq_handlers
| Inputs:  Nothing
| Returns:          Nothing
| Purpose:          Set the pointers in the interrupt table to point to
|                   our funtions ie. setup for ISR's.
|---------------------------------------------------------------------*/
void init_irq_handlers(void)
{
        _disable();
                old_IRQ1_handler = _dos_getvect(IRQ1_VECTOR + 8);
                _dos_setvect(IRQ1_VECTOR + 8, new_IRQ1_handler);
                Gi_old_mask = inp(0x21);
                outp(0x21,Gi_old_mask & ~(1 << IRQ1_VECTOR));
        _enable();
}

|/*---------------------------------------------------------------------
| Function:         restore do this before exiting program
| Inputs:  Nothing
| Returns:          Nothing
| Purpose:          Restore interrupt vector table.
|---------------------------------------------------------------------*/
void restore(void)
{
        /* Restore the old vectors */

        _disable();

                _dos_setvect(IRQ1_VECTOR + 8, old_IRQ1_handler);
                outp(0x21,Gi_old_mask);

        _enable();


}
```

ECAN527HR Specifications

Host Interface

Memory mapped into low memory, occupies 256 bytes

Jumper-selectable base address, 11 jumpers

8-bit data bus, 16 bit PC/104 bus connector

Jumper selectable XT and AT interrupts

CAN Interface

-Galvanically isolated transceiver with 1 Mb/s datarate

-Timing parameters and speed of bus programmable

-Balanced CAN-bus choke for low EMI emissions

-Jumper selectable 120 Ohm onboard termination resistor

-0.8W 5V isolated output power for other field devices

-Fiberoptic transmitter and receiver 1Mb/s datarate

using the RTD CAN Spider active hub on ECAN527HR-2 boards

Digital I/O (non isolated)

Number of lines	**6**
LED's	**2**
Series resistor	**100 Ohms**
Pull-up / Pull-down	**Bits 1-3 10K pull-down**

Connectors

Galvanically isolated CAN bus	**ISO 11898 compliant**
Fiberoptic CAN bus	**HP Versa-Link connectors**
Non Isolated I/O	**Header connector 10-Pin**

Electrical

Operating temperature range	**-40 to +85C**
Supply voltage	**+5V only**
Power consuption	**115mA**

CE

Meets CE approval for "Heavy Industrial Equipment" in RTD IDAN frame

Chapter 5 RETURN POLICY AND WARRANTY

Return Policy

If you wish to return a product to the factory for service, please follow this procedure:

Read the Limited Warranty to familiarize yourself with our warranty policy.

Contact the factory for a Return Merchandise Authorization (RMA) number.

Please have the following available:

- Complete board name
- Board serial number
- A detailed description of the board's behavior

List the name of a contact person, familiar with technical details of the problem or situation, **along with their phone and fax numbers, address, and e-mail address** (if available).

List your shipping address!!

Indicate the shipping method you would like used to return the product to you.
We will not ship by next-day service without your pre-approval.

Carefully package the product, using proper anti-static packaging.

Write the RMA number in large (1") letters on the outside of the package.

Return the package to:

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA

Limited Warranty

RTD Embedded Technologies, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from RTD Embedded Technologies, INC. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, RTD Embedded Technologies will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to RTD Embedded Technologies. All replaced parts and products become the property of RTD Embedded Technologies. Before returning any product for repair, customers are required to contact the factory for an RMA number.

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by RTD Embedded Technologies, "acts of God" or other contingencies beyond the control of RTD Embedded Technologies), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN RTD Embedded Technologies. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND RTD Embedded Technologies EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL RTD Embedded Technologies BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

RTD Embedded Technologies, Inc.
103 Innovation Blvd.
State College PA 16803-0906
USA
Our website: www.rtd.com

NOTES:

Base Address: _____

IRQ selection: _____